PR #2587 完整报告

radixark/miles

openenv: stop the episode at a length-truncated turn

合并时间：2026-08-18 08:07

原文链接：http://prhub.com.cn/radixark/miles/pull/2587

执行摘要

- 一句话：截断轮次立即结束 episode，消除 22.7% 无效生成时间
- 推荐动作：值得精读。代码量极小（核心仅 3 行），但问题定位方法极具参考价值：用实测数据（6469 episodes、144.7h/638h、单样本 73% 时间被丢弃）量化不可见浪费，通过 dashboard 渲染盲区反推根因，再以训练端丢弃边界校准 rollout 端生成边界。设计上 " 截断即停止、但保留评分 " 的取舍清晰、与训练契约严格对齐，是实现最小修复的范例。

功能与动机

在 tbench2 agent loop 中，命中 --rollout-max-response-len 的轮次会以 finish_reason="length" 返回并关闭可训练样本，但 agent loop 不知情：它会执行被截断的命令并继续生成直到 OPENENV_MAX_TURNS 或 episode 墙钟上限，这些 token 训练端全会丢弃。PR body 给出实测数据：在 GB300 16 节点 GLM-5.2 运行的 6469 个 episodes 中，截断后的 ghost tail 消耗了全部 episode 墙钟时间的 22.7%（638 小时中的 144.7 小时），且集中于最长上下文；垂到墙钟上限的样本以 ABORTED 结束，还会通过 check_no_aborted 拖垮整组。这段尾巴在 dashboard 的 batch anatomy 中不可见，渲染为长 "queue/waiting" 带，因此一直未被察觉。

实现拆解

1. 变更入口：examples/experimental/openenv/openenv_agent_function.py 的 run_episode() 内部 body() 闭包的 turn 循环（while turns < max_turns）。这是 openenv agentic 路径发起策略调用、执行命令、回填对话的唯一入口。
2. 核心逻辑改造：原先直接取 completion.choices[0].message，现先绑定 choice = completion.choices[0] 再取 message，并在 convo.append(message.model_dump(exclude_none=True)) 之后、命令解析之前插入 if choice.finish_reason == "length": break。该位置经过精心选择：assistant 消息已回显给会话服务器（保持前缀匹配），但截断轮次的残缺命令不会被执行，也不会被拼入下一轮请求，循环直接跳出。
3. 评分保留：跳出循环后仍走原有的 env.step(action_cls(action_type="evaluate")) 评分逻辑，episode 按截断时的现状打分，reward 归一逻辑不变。
4. 测试配套：tests/fast/examples/experimental/openenv/test_openenv_agent_function.py 三处改动——(a) 既有 _FakePolicy._create 的返回补上 finish_reason="stop"，让 mock 与真实 SDK 响应结构一致；(b) 新增 _TruncatedPolicy，每轮返回 finish_reason="length" 且命令在 bash fence 内被腰斩；(c) 新增

test_truncated_turn_ends_the_episode，断言只发生一次 policy 调用、cut-off 命令不执行、evaluate 仍运行、turns == 1。

5. 无额外配套：本 PR 不涉及配置、schema、部署或文档改动，全部变更集中在上述 2 个文件。

关键文件：

- examples/experimental/openenv/openenv_agent_function.py（模块 开放环境；类别 source；类型 core-logic；符号 run_episode）：核心修复所在：turn 循环中新增 finish_reason=="length" 提前退出，并跳过截断命令的执行，改动虽小但直接消除 22.7% 的 episode 墙钟浪费。
- tests/fast/examples/experimental/openenv/test_openenv_agent_function.py（模块 开放环境；类别 test；类型 test-coverage；符号 _TruncatedPolicy, init, _create, test_truncated_turn_ends_the_episode）：新增 _TruncatedPolicy 与 test_truncated_turn_ends_the_episode，验证截断轮次后循环停止、命令不执行、评分照常，是整个行为契约的回归防线。

关键符号：run_episode, body, _TruncatedPolicy._create, test_truncated_turn_ends_the_episode

关键源码片段

examples/experimental/openenv/openenv_agent_function.py

核心修复所在：turn 循环中新增 finish_reason=="length" 提前退出，并跳过截断命令的执行，改动虽小但直接消除 22.7% 的 episode 墙钟浪费。

```
turns = 0
while turns < max_turns:
    turns += 1

    # 请求策略模型生成下一轮动作；真实 SGLang 响应可能因
    # --rollout-max-response-len 被截断，此时 finish_reason 为 "length"。
    completion = await policy.chat.completions.create(
        model=model_name, messages=convo, extra_body=request_kwargs
    )
    choice = completion.choices[0] # 一次性绑定，避免重复索引
    message = choice.message
    reply = message.content or ""

    # 回显 assistant turn：必须原样保留 SDK 解析出的 message，否则
    # 会话服务无法按前缀匹配，会触发 "rollback failed" 类错误。
    convo.append(message.model_dump(exclude_none=True))

    # 核心修复：训练端在 finish_reason == "length" 处关闭样本，采集
    # 不保留截断之后的任何内容。rollout 端若继续循环，后续所有轮次
    # 都是训练将丢弃的 token，白白消耗引擎算力。
    if choice.finish_reason == "length":
        break
```

```
            # 只有完整命令才被解析执行；截断轮次的残缺命令（mid-stream
            # 垃圾）在此被跳过，不再 env.step(exec)。
            command = _strip_fence(reply) if "```" in reply else reply.strip()
            if not command or command.upper().startswith("TASK_COMPLETE"):
                break

            step_result = await env.step(action_cls(action_type="exec", command=command))
            output = _obs_field(step_result, "output")
            content = output[:_OBS_CHAR_CAP] or "(no output)"
            convo.append({"role": "user", "content": content})

        # 无论正常结束还是截断结束，评分照常运行：episode 按现状打分，
        # 训练端据此得到 reward（无权威 verdict 时为 None，样本被丢弃）。
        eval_result = await env.step(action_cls(action_type="evaluate"))
```

tests/fast/examples/experimental/openenv/test_openenv_agent_function.py

新增 _TruncatedPolicy 与 test_truncated_turn_ends_the_episode，验证截断轮次后循环停止、命令不执行、评分照常，是整个行为契约的回归防线。

```
class _TruncatedPolicy:
    """模拟真实 SDK 响应：每轮都返回被 per-turn 上限截断的命令。"""

    def __init__(self):
        self.n = 0 # 记录 policy 被调用次数，用于断言循环提前终止
        self.chat = types.SimpleNamespace(completions=types.SimpleNamespace(create=self._
        create))

    async def _create(self, **kw):
        self.n += 1
        # 命令在 "make -j && ./run_all_the" 处被截断，bash fence 未闭合
        text = "```bash\nmake -j && ./run_all_the"
        msg = types.SimpleNamespace(
            content=text, model_dump=lambda exclude_none=True: {"role": "assistant", "content":
            text}
        )
        # 关键：finish_reason 必须与真实 SGLang 响应一致，才能触发截断分支
        return types.SimpleNamespace(choices=[types.SimpleNamespace(message=msg, finish_
        reason="length")])


def test_truncated_turn_ends_the_episode(monkeypatch):
    """finish_reason="length" 的轮次必须立即结束 episode：
    不执行残缺命令、不发起第二次 policy 调用、evaluate 照常运行、turns == 1。"""
    monkeypatch.setattr(oaf, "load_tbench2", lambda: _CLASSES)

    async def spying_with_env(env_cls, env_url, body):
        return await body(env_cls())

    monkeypatch.setattr(oaf, "_with_env", spying_with_env)
```

```
policy = _TruncatedPolicy()
reward, metrics = run_async(
    oaf.run_episode(policy, "m", [{"role": "system", "content": "s"}], {}, {"task_id": "t1"})
)
assert policy.n == 1, "the loop must stop at the truncated turn"
actions = _FakeEnv.last_actions
execs = [a for a in actions if a.action_type == "exec"]
assert all("/tmp/tbench2_env_runs" in (a.command or "") for a in execs), execs
assert any(a.action_type == "evaluate" for a in actions), "scoring still runs"
assert reward == 1.0 and metrics["turns"] == 1
```

评论区精华

唯一一条实质 review 评论来自 nblintao，针对第一版提交中 `if completion.choices[0].finish_reason == "length":` 的写法留了 TODO，建议将 `completion.choices[0]` 绑定为局部变量，避免条件判断里重复索引。作者随后提交 "Bind choices[0] once"（f5654ac）将 `choice` 绑定一次并复用，评论得到落实，nblintao 最终 APPROVED。除此之外没有其他争议或设计交锋，PR 主体一次性通过评审。

- completion.choices[0] 重复索引与局部绑定 (style): 作者随后提交 "Bind choices[0] once"（f5654ac），改为先绑定 choice 再复用，落实评论建议，nblintao 最终 APPROVED。

风险与影响

- 风险：
  1. 行为语义变更：截断轮次原本会执行被截断的命令（可能是 mid-stream 垃圾），现在直接跳过执行并结束 episode，环境状态与先前不同，最后 evaluate 的评分可能因此变化。但由于训练端本来就不保留截断后的任何轮次，这一变化影响的只是评分时点的环境状态，方向上是合理的。
  2. 依赖 finish_reason 字段语义：修复依赖 SDK/ 服务端在超长生成时确实返回 "length"。若某条路径返回空字符串或 "stop"，修复不生效但也无副作用；本 PR 未处理 finish_reason 为 None 的防御场景。
  3. 测试覆盖深度：103 个 fast 测试均基于 fake policy 模拟响应，未覆盖真实 SGLang 端到端路径；真实 SDK 的 choices[0].finish_reason 字段结构假设仅由 _FakePolicy 模拟验证。
  4. 波及范围受限：改动仅在 examples/experimental/openenv/ 下，核心库的 rollout、session 逻辑未受影响，回归面小。 - 影响：对运营成本，22.7% 的 episode 墙钟浪费被消除，长上下文场景（ghost tail 集中区）的引擎计算收益更高；对稳定性，垂到墙钟上限的样本不再产生 ABORTED，check_no_aborted 拖垮整组的连锁风险显著下降，fully-async 全组 staleness 尾部也会因 episode 缩短而变薄；对可观测性，batch anatomy 中原本渲染为长 "queue/waiting" 带的不可见尾巴消失，dashboard 数据更接近真实工作负载；对团队，这是 " 训练端采样边界与 rollout 端生成边界不对齐 " 的典型案例，可作为后续 agentic 路径评审的契约核对参照。 - 风险标记：行为语义变更：截断轮次不再执行命令，依赖 finish_reason 字段语义，评分时环境状态可能变化，仅覆盖 fake policy 的 fast 测试

关联脉络

- PR #2544 Do not kill the run when one sample's collect_samples loses its connection: 同属 " 单个样本的边界事件不应拖垮整体运行 " 的健壮性修复方向：本 PR 处理截断导致 ABORTED 组的问题，PR 2544 处理采集断连的问题，均在 rollout/session 层面对齐训练端契约。
- PR #2558 test(ci): move the agentic-env integration tests under tests/fast so CI runs them: 该 PR 将 tests/fast/examples/experimental/openenv 纳入 CI 运行，本 PR 的截断回归测试恰好落在此目录，依托其 CI 覆盖。
- PR #2545 Reuse one Daytona client per process instead of one per create attempt: 同为 openenv 运行效率与稳定性修复（fd 泄漏导致训练停止），与本 PR 在 openenv 沙箱 /agent 基础设施侧形成持续改进脉络。